Practical Privacy-Preserving Location-Sharing Based Services with Aggregate Statistics

Michael Herrmann
KU Leuven ESAT/COSIC, iMinds
Leuven, Belgium
michael.herrmann@esat.kuleuven.be

Alfredo Rial
IBM Research
Rüschlikon, Switzerland
lia@zurich.ibm.com

Claudia Diaz
KU Leuven ESAT/COSIC, iMinds
Leuven, Belgium
claudia.diaz@esat.kuleuven.be

Bart Preneel
KU Leuven ESAT/COSIC, iMinds
Leuven, Belgium
bart.preneel@esat.kuleuven.be

ABSTRACT

Location-sharing-based services (LSBSs) allow users to share their location with their friends in a sporadic manner. In currently deployed LSBSs users must disclose their location to the service provider in order to share it with their friends. This default disclosure of location data introduces privacy risks. We define the security properties that a privacy-preserving LSBS should fulfill and propose two constructions. First, a construction based on identity based broadcast encryption (IBBE) in which the service provider does not learn the user's location, but learns which other users are allowed to receive a location update. Second, a construction based on anonymous IBBE in which the service provider does not learn the latter either. As advantages with respect to previous work, in our schemes the LSBS provider does not need to perform any operations to compute the reply to a location data request, but only needs to forward IBBE ciphertexts to the receivers. We implement both constructions and present a performance analysis that shows their practicality. Furthermore, we extend our schemes such that the service provider, performing some verification work, is able to collect privacy-preserving aggregate statistics on the locations users share with each other.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: Electronic Commerce–Distributed commercial transactions; Security

Keywords

Location Privacy; Broadcast Encryption; Vector Commitments

WiSec'14, July 23–25, 2014, Oxford, UK.

ACM 978-1-4503-2972-9/14/07 ...$15.00.
http://dx.doi.org/10.1145/2627393.2627414 .

1. INTRODUCTION

The emergence of mobile electronic devices with positioning capabilities (e.g. through the Global Positioning System, GPS), such as smart phones and tablet computers, has fostered the appearance of a wide variety of Location Based Services (LBSs). With these services, users can find nearby places of interest, share their location with friends, and obtain information about their surroundings.

Location-sharing-based services (LSBSs) permit users to share their current location or activity with other people. The shared location data may be in the form of GPS coordinates, although in GeoSocial Networks (GSN), such as *Foursquare* and *Facebook-check-in*, it is common that users announce their location in a more socially meaningful way by providing the venue (e.g., name of the restaurant) at which they are currently present. The action is commonly referred to as *check-in*. Every day millions of users enjoy GSN and share millions of locations with each other.[1]

While LSBSs are indeed useful, the disclosure of location data raises significant privacy concerns. Service providers and other third parties with access to accurate location data can infer private user information, such as their movement patterns, home address, lifestyle and interests [25]. Further, making these inferences is even easier if users share the venue rather than just submitting coordinates, as any uncertainties introduced by possible inaccuracies in the GPS coordinates are removed. We note that, although GSNs offer configurable privacy settings [27], they are still privacy invasive, as the LSBS provider learns the users' location regardless of the privacy settings.

Location Privacy Preserving Mechanisms (LPPMs) that implement obfuscation strategies, such as adding dummy locations [35] or reducing precision [31], are unsuitable for LSBS. This is because, when transmitting an obfuscated location to the service provider, the service provider naturally is only able to forward this obfuscated location to the user's friends. This conflicts with the main functionality of LSBSs. Therefore, LPPMs have been proposed in which users share keys allowing them to encrypt and decrypt their location data [22, 45]. In those solutions, the LSBS provider stores encrypted location data and computes the reply for a user

[1] `https://foursquare.com/about/`
`http://www.socialbakers.com/blog/167-interesting-facebook-places-numbers`

requesting location data of her friends. A provider offering such an LSBS is unable to learn statistics about its users' whereabouts. Consequently, this renders the common business model of offering a free service in exchange for the users' data impossible. An alternative is to offer a paid service, but this might only be feasible if the fees are sufficiently low.

In this paper we propose two schemes based on identity-based broadcast encryption [20]. The first protocol reveals the identities of the friends that should receive location information to the LSBS provider and also to the other receivers of that location information. In the second protocol, those identities are hidden towards the service provider as well as towards other users (including the receivers of the location update), thanks to the use of anonymous identity-based broadcast encryption. The advantage over existing work is that in our schemes the LSBS provider does not need to perform complex operations in order to compute a reply for a location data request, but only needs to forward data. This reduces the cost of an LSBS provider that is then able to offer its service for a lower price if pursuing a subscription-based business model. Furthermore, we extend our schemes such that the service provider is able to collect privacy-preserving statistics on the locations shared by the users. This extension does require the LSBS provider to perform additional computations. The obtained statistics could be monetized to compensate for this additional overhead or to facilitate a free service.

We have implemented both schemes on a *Samsung S III mini* smart phone and provide results on the computation time, bandwidth overhead and energy consumption. Our evaluation shows that the performance of the first scheme is independent of the number of users in the system. Furthermore, it imposes minimal computational and bandwidth overhead on both, the LSBS provider and the users' mobile devices. In the second scheme a user is able to choose a trade-off between privacy, computation and bandwidth overhead. We study this trade-off and provide recommendations to increase the level of privacy for the same amount of resources.

The remainder of this paper is structured as follows. In Section 2, we review previous work on privacy-preserving location-based services and argue that none of the existing approaches is suitable for implementing privacy-preserving LSBSs. We define privacy-preserving LSBS in Section 3. In Section 4, we introduce our two schemes. We provide a detailed performance analysis showing the feasibility of our approach in Section 5. In Section 6, we extend our schemes to allow for aggregate statistics collection. We discuss our approach and results in Section 7. Finally, in Section 8 we conclude our work.

2. RELATED WORK

In this section we review obfuscation-based LPPMs and argue that they are not suitable for protecting location privacy in LSBS. Subsequently, we review LPPMs that rely on cryptographic primitives. Some of them have been deliberately designed for protecting location privacy in LSBS; others have a more general purpose. Our evaluation shows that obfuscation-based LPPMs are not suitable for privacy-preserving LSBS and that our system has several advantages over existing privacy-preserving LSBS.

Other works have examined location privacy in GSNs considering a different threat model. Gambs et al. [27] and Vicente et al. [48] review several GSNs and analyze their privacy issues. However, in their privacy evaluation, they do not consider it to be a privacy breach if the service provider learns the user locations. An analysis of the inferences that can be made about users based on where they check-in while using Foursquare is provided by Pontes et al. [43].

2.1 Obfuscation-based LPPMs

While works like [22, 45] have already noted that LPPMs based on anonymization and precision-reduction are not suitable to protect location privacy in LSBS, we provide here a more detailed evaluation. We therefore follow the categorization in [47] which distinguishes between these four obfuscation strategies: *location hiding*, *perturbation*, *adding dummy regions*, and *reducing precision*. In the following we argue that none of these obfuscation-based LPPMs are applicable to protect location privacy in LSBSs. We therefore consider the following LSBS application scenario: A user A is currently at one of her favorite locations and wishes to share this information with her friends. This could be either because A simply wants to inform her friends, or to enable them to join her at this place.

The *location hiding* strategy [4] consists of not sending the location data to the LBS and is thus impractical. In this case user A would not be able to share her location with her friends. Some LPPMs propose to change pseudonyms after a period of location hiding [32, 37]. However, this is also impractical, because the check-in is supposed to be received by the same set of friends and therefore identifies user A. LPPMs that rely on *perturbation* submit a location different from A's actual location [34]. As a rather inconvenient result, the user's friends learn a wrong place and if they decide to join their friend, they would realize upon arriving that A is actually not present there. The *adding dummy regions* strategy [35] consists of submitting fake locations along with the user's actual location. In this case A would check-in at several places and her friends could not distinguish real from fake check-ins. Finally, with the *reducing precision* strategy [31] A would send a *cloak region* that contains her current location, but she would not reveal her precise whereabouts, making it extremely difficult for her friends to find her. We note that with all obfuscation strategies, users could rely on personal contact in order to obtain A's precise location after learning the obfuscated location. However, such a system would have significant usability issues. The key limitation of these techniques is that they do not make a distinction between information revealed to friends and to the service provider. Thus, in order to protect their location information towards the service provider, users must lower the quality of location information shared with their friends.

2.2 Cryptographic LPPMs

Freudiger [26] proposes that users should use symmetric or asymmetric encryption and use the service provider solely as a rendez-vous point to exchange encrypted data. Longitude by Dong and Dulay [22] propose to use proxy-encryption, which guarantees that the service provider is not able to learn the location update and, furthermore, that the ciphertext can be modified by the service provider such that only intended receivers are able to successfully decrypt. Puttaswamy et al. [45] propose a scheme which combines encryption with location transformation in order to build a

location-based application, such as privacy-preserving LSBS. As already mentioned in Section 1 the proposed LPPMs impose a computational overhead at the LSBS provider, which makes offering such a service more expensive. In our schemes the cost for the LSBS provider is kept at a minimum in order to make running such a service as cheap as possible. Furthermore, the service provider can decide to engage in additional computation overhead and therefore obtain statistics about its users' whereabouts. We note that this overhead is kept low since the service provider only needs to forward data and verify zero-knowledge proofs, whose cost can be reduced using batch verification. Note however that our scheme, in contrast to the works mentioned above, introduces a trusted key generation center.

Carbunar et al. [16] propose privacy preserving protocols for badge and mayor applications in GSNs. While this is closely related to our work, their scheme does not allow users to exchange their locations.

In privacy-preserving friend nearby notification, users can privately learn whether a friend is in close proximity. Such a service can be realized by homomorphic encryption [50], private equality testing [41] and private threshold set intersection [39]. These protocols are different from our solution, because in privacy-preserving location sharing protocols, location updates are sent to friends regardless of their current location, i.e. regardless of how close they are.

Bilogrevic et al. [5] propose two protocols to allow users to compute a fair rendez-vous point in a privacy preserving manner. This differs from our work in that we focus on location sharing, and not on deciding on where to meet after a group of users has deliberately decided to do so.

Popa et al. [44] propose a privacy-preserving protocol to compute aggregate statistics from users' location updates. However, in this protocol users do not share their location with other users.

Finally, some works employ Private Information Retrieval (PIR) so that the users retrieve information (e.g., points of interest) related to their surroundings [30, 42]. PIR could in principle be employed to build privacy-preserving LSBS. However, PIR operations are rather costly at the service provider side which we again argue that introduces intolerable overhead for a service provider.

3. DEFINITION OF PRIVACY FOR LSBS

Our LSBS involves the following parties: a key generation center, a service provider $\mathcal{P}$ and a set of users $\mathcal{U}_i$ for $i = 1$ to n. Figure 1 shows the parties in the system.

A privacy-invasive protocol that realizes the desired functionality works as follows. A user $\mathcal{U}_i$ sends a message to the service provider that indicates the place *loc* that $\mathcal{U}_i$ wishes to share, and the set $S \subseteq [1, n]$ of users $\mathcal{U}_j$ $(j \in S)$ that should learn that $\mathcal{U}_i$ shares *loc*. Then the service provider forwards to users $\mathcal{U}_j \in S$ the message $(\mathcal{U}_i, loc)$ to inform them that $\mathcal{U}_i$ has shared *loc*. As can be seen, this protocol is privacy-invasive. The service provider learns the location *loc* that $\mathcal{U}_i$ shares, and the identities of the users $\mathcal{U}_j$ $(j \in S)$. The privacy properties that our LSBS should fulfill are the following:

Sender Privacy. No coalition of users $\mathcal{U} \notin S$ and service provider $\mathcal{P}$ should learn any information on *loc*.

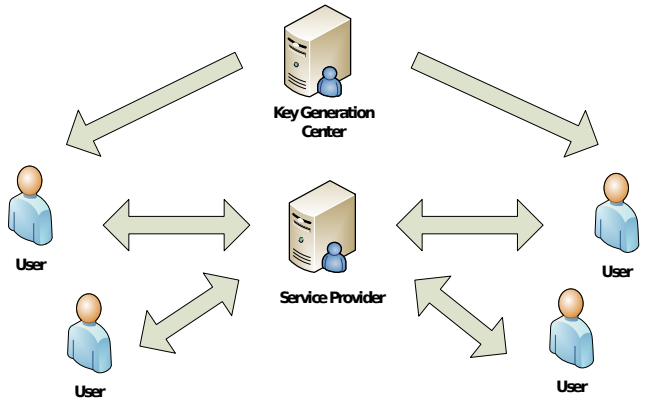


Figure 1: System Model of a privacy-preserving LSBS. The key generation center sets up the public parameters and provides users with secret keys. The service provider transfers messages between users.

Receiver Privacy. No coalition of users $\mathcal{U}_j$ such that $j \neq \{i, j^*\}$ and service provider $\mathcal{P}$ should learn any information on whether $j^* \in S$ or $j^* \notin S$.

Our schemes in Section 4 are secure against active adversaries, i.e., adversaries that deviate in any possible way form the protocol execution. The security of our schemes relies on the security of identity based broadcast encryption. The key generation center is trusted, which is an assumption that every identity-based cryptographic scheme makes.

4. CONSTRUCTIONS OF LSBS

Our schemes are based on identity-based broadcast encryption, which we describe in Section 4.1. In Section 4.2, we describe the *sender private* scheme, which fulfills the sender privacy property. In Section 4.3, we describe the *fully private* scheme, which fulfills both the sender privacy and the receiver privacy properties.

4.1 Identity-Based Broadcast Encryption

Broadcast encryption allows a sender to encrypt a message m to a set of receivers $S \in [1, n]$, so that no coalition of receivers not in S can decrypt. A broadcast encryption scheme consists of the following algorithms:

$\mathsf{Setup}(1^\lambda, n, \ell)$**.** On input the number of users n, the security parameter 1^λ, and the maximum size $\ell \leq n$ of a broadcast recipient group, output the public key pk and the secret key sk.

$\mathsf{KeyGen}(i, sk)$**.** On input an index i and the secret key sk, output a secret key d_i for user $\mathcal{U}_i$.

$\mathsf{Enc}(pk, S, m)$**.** On input the recipient group $S \in [1, n]$, the public key pk and the message m, output the ciphertext c.

$\mathsf{Dec}(pk, d_i, c)$**.** On input the public key pk, the secret key d_i of user $\mathcal{U}_i$ and a ciphertext c, output m if $i \in S$ or else the failure symbol $\perp$.

In IBBE, a trusted key generation center $\mathcal{KGC}$ creates the parameters and computes the secret keys of the receivers. Note that the secret key sk allows the decryption of every ciphertext. If ciphertexts c do not reveal the set of receivers S, the broadcast encryption scheme is anonymous.

4.2 Sender-Private LSBS

Our sender-private LSBS (SPLS) uses an IBBE scheme that is not anonymous. In such a scheme, ciphertexts c contain a description of the recipient group S. Our scheme works as follows:

Setup Phase. $\mathcal{KGC}$ executes the setup algorithm $\mathsf{Setup}(1^\lambda, n, \ell)$ on input the security parameter 1^λ, the number of users n and the maximum size $\ell \leq n$, publishes the public key pk and stores the secret key sk. Users obtain pk.

Registration Phase. Each user $\mathcal{U}_i$ registers with the service provider by sending the index i. Additionally, $\mathcal{U}_i$ receives the key d_i from $\mathcal{KGC}$, which runs $\mathsf{KeyGen}(i, sk)$.

Main Phase. To share a location loc, a user $\mathcal{U}_i$ runs $c \leftarrow \mathsf{Enc}(pk, S, i||loc)$ and sends c to the service provider $\mathcal{P}$. $\mathcal{P}$ gets S from c and sends c to the users $\mathcal{U}_j$ $(j \in S)$. Each user $\mathcal{U}_j$ runs $\mathsf{Dec}(pk, d_j, c)$ to output the message $i||loc$.

We note that the registration and main phases can be interleaved, i.e., users can join our SPLS dynamically.

Our scheme fulfills the sender privacy property. The IBBE scheme ensures that no coalition of service provider $\mathcal{P}$ and users $\mathcal{U} \notin S$ can decrypt a ciphertext c computed on input S. However, this scheme does not fulfill the receiver privacy property. Since the IBBE scheme is not anonymous, the ciphertext c reveals the identity of the receivers $\mathcal{U}_j$ $(j \in S)$.

4.2.1 Construction of IBBE

In Section 5, we instantiate our SPLS with a broadcast encryption scheme in order to implement it and evaluate its performance. Broadcast encryption was first formalized by Fiat and Naor [24]. The first fully collusion secure broadcast schemes, i.e., schemes where security holds even if all the users not in the recipient group S collude, were described in [40]. The first public key broadcast encryption scheme was proposed in [21].

In the schemes mentioned above, the size of the ciphertext grows linearly with the size of the recipient group. Boneh, Gentry and Waters [8] proposed the first schemes with constant size ciphertexts. Their schemes have selective security, i.e., the adversary should decide the target recipient group to be attacked before the setup phase. Identity-based broadcast encryption was proposed in [20], which describes also selectively secure schemes.

Broadcast encryption and identity-based broadcast encryption with adaptive security was first achieved in [28]. These schemes achieve constant size ciphertexts in the random oracle model and under q-based assumptions. In [49] and [36], broadcast encryption schemes secure under static assumptions are proposed. In [36], an identity-based broadcast encryption scheme secure under static assumptions is also proposed, but it only achieves selective security.

The main property of identity-based broadcast encryption is that the scheme is efficient when the total number of users n is exponential in the security parameter 1^λ. Since our SPLS could have millions of users, schemes that, despite having constant size ciphertexts, have public key or user secret keys of size that grows linearly with n are less suitable. Therefore, we instantiate our SPLS with the adaptively secure identity-based broadcast encryption in [28]. This scheme, which is secure in the random oracle model, in addition to constant size ciphertexts, has a public key of size independent of n that grows linearly with ℓ and user decryption keys d_i of constant size. The scheme in [28] employs bilinear maps.

Bilinear maps.

Let $\mathbb{G}$, $\tilde{\mathbb{G}}$ and $\mathbb{G}_t$ be groups of prime order p. A map $e : \mathbb{G} \times \tilde{\mathbb{G}} \to \mathbb{G}_t$ must satisfy bilinearity, i.e., $e(g^x, \tilde{g}^y) = e(g, \tilde{g})^{xy}$; non-degeneracy, i.e., for all generators $g \in \mathbb{G}$ and $\tilde{g} \in \tilde{\mathbb{G}}$, $e(g, \tilde{g})$ generates $\mathbb{G}_t$; and efficiency, i.e., there exists an efficient algorithm $\mathsf{BMGen}(1^k)$ that outputs the pairing group setup $(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_t, e, g, \tilde{g})$ and an efficient algorithm to compute $e(a, b)$ for any $a \in \mathbb{G}$, $b \in \tilde{\mathbb{G}}$. If $\mathbb{G} = \tilde{\mathbb{G}}$ the map is symmetric and otherwise asymmetric.

Let (E, D) be a secure symmetric key encryption scheme. The scheme in [28] works as follows:

$\mathsf{Setup}(1^\lambda, n, \ell)$**.** On input the number of users n, the security parameter 1^λ, and the maximum size $\ell \leq n$ of a broadcast recipient group, run $(p, \mathbb{G}, \mathbb{G}_t, e) \leftarrow \mathsf{BMGen}(1^\lambda)$. Set $g_1, g_2 \leftarrow \mathbb{G}$. Set $\alpha, \beta, \gamma \leftarrow \mathbb{Z}_p$. Set $\hat{g}_1 \leftarrow g_1^\beta$ and $\hat{g}_2 \leftarrow g_2^\beta$. Set $pk = (p, \mathbb{G}, \mathbb{G}_t, e, n, \ell, g_1^\gamma, g_1^{\gamma \cdot \alpha}, \{g_1^{\alpha^j}, \hat{g}_1^{\alpha^j}, g_2^{\alpha^k}, \hat{g}_2^{\alpha^k} : j \in [0, \ell], k \in [0, \ell-2]\})$. Output pk and the secret key $sk = (\alpha, \gamma)$.

$\mathsf{KeyGen}(i, sk)$**.** On input an index i and the secret key sk, pick random $r_i \leftarrow \mathbb{Z}_p$ and output

$$d_i = (i, r_i, h_i = g_2^{\frac{\gamma - r_i}{\alpha - i}})$$

$\mathsf{Enc}(pk, S, m)$**.** On input the recipient group $S \in [1, n]$, the public key pk and the message m, set $\tau \leftarrow \{0,1\}^{\mathcal{O}(\lambda)}$. Set $F(x)$ as the $(\ell-1)$-degree polynomial that interpolates $F(i) = H(\tau, i)$ for $i \in S$ and $F(i) = 1$ for $i \in [n+j]$ with $j \in [k+1, \ell]$, where $H : \{0,1\}^{\mathcal{O}(\lambda)} \times [1, n] \to \mathbb{Z}_p$ is a hash function modelled as a random oracle. Set $k = |S|$ and parse S as $\{i_1, \ldots, i_k\}$. Set $i_j \leftarrow n + j$ for $j \in [k+1, \ell]$. Set $P(x) = \prod_{j=1}^{\ell}(x - i_j)$. Set $t \leftarrow \mathbb{Z}_p$ and set $K \leftarrow e(g_1, \hat{g}_2)^{\gamma \cdot \alpha^{\ell-1} \cdot t}$. Set $\mathrm{Hdr} \leftarrow \langle C_1, \ldots, C_4 \rangle = \langle \hat{g}_1^{P(\alpha) \cdot t}, g_1^{\gamma \cdot t}, g_1^{F(\alpha) \cdot t}, e(g_1, \hat{g}_2)^{\alpha^{\ell-1} \cdot F(\alpha) \cdot t} \rangle$. Compute $C \leftarrow \mathsf{E}(K, m)$ and output $c = (\tau, \mathrm{Hdr}, C, S)$.

$\mathsf{Dec}(pk, d_i, c)$**.** On input the public key pk, the secret key d_i and a ciphertext c, parse d_i as $\langle i, r_i, h_i \rangle$, c as $(\tau, \mathrm{Hdr}, C, S)$ and Hdr as $\langle C_1, \ldots, C_4 \rangle$. Define $P(x)$ as above and compute $F(x)$ from τ as above. Set

$$P_i(x) = x^{\ell-1} - \frac{P(x)}{(x-i)},\ F_i(x) = \frac{F(x) - F(i)}{(x-i)},$$

$$\text{and } e_i = -\frac{r_i}{F(i)}$$

and

$$K \leftarrow e(C_1, h_i \cdot g_2^{e_i \cdot F_i(\alpha)}) \cdot e(C_2 \cdot C_3^{e_i}, \hat{g}_2^{P_i(\alpha)}) / C_4^{e_i}.$$

Output $m \leftarrow \mathrm{D}(K, C)$.

We note that a user of LSBS usually shares her location with the same recipient group, i.e., with her friends. Therefore, broadcast encryption is used to share a symmetric key with that recipient group, and messages are encrypted using an efficient symmetric key encryption scheme. Broadcast encryption is used again only when the recipient group changes

or when the symmetric key should be renewed for security reasons.

4.3 Fully-Private LSBS

Our fully-private LSBS (FPLS) uses an anonymous IBBE scheme. In such scheme, ciphertexts c do not reveal the recipient group. The setup and registration phases of this scheme work as the ones described in Section 4.2. The main phase works as follows:

Main Phase. To share a location loc, a user $\mathcal{U}_i$ runs $c \leftarrow \mathsf{Enc}(pk, S, i||loc)$ and sends c to the service provider $\mathcal{P}$. $\mathcal{P}$ forwards c to every user $\mathcal{U}_j$ such that $j \neq i$. Each user $\mathcal{U}_j$ runs $\mathsf{Dec}(pk, d_j, c)$ to get either the message $i||loc$ or $\perp$.

As in the construction in Section 4.2, this scheme fulfills the sender-private property. Additionally, this scheme fulfills the receiver privacy property. Since the IBBE scheme is anonymous, the ciphertext c does not reveal the identity of the receivers $\mathcal{U}_j$ $(j \in S)$.

This construction requires location updates to be broadcast to all users. Therefore, we propose a variant that allows to trade-off communication efficiency and location-privacy. In this variant, the map is divided into regions $\text{reg}_1, \ldots, \text{reg}_r$ and users reveal to the service provider the region where they are located and the region from where they would like to receive location updates.

Region Phase. A user $\mathcal{U}_i$ sends to the service provider the region to which location updates she wishes to receive should be associated.

Main Phase. To share a location $loc \in$ reg, a user $\mathcal{U}_i$ runs $c \leftarrow \mathsf{Enc}(pk, S, i||loc)$ and sends (c, reg) to the service provider $\mathcal{P}$. $\mathcal{P}$ forwards c to every user $\mathcal{U}_j$ such that $j \neq i$ and reg equals the region sent by $\mathcal{U}_j$ in the Region Phase. Each user $\mathcal{U}_j$ runs $\mathsf{Dec}(pk, d_j, c)$ to get either the message $i||loc$ or $\perp$.

4.3.1 Construction of Anonymous IBBE

In Section 5, we instantiate our FPLS with an anonymous broadcast encryption scheme in order to implement it and evaluate its performance. Barth et al. [3] propose an anonymous broadcast encryption scheme secure in the random oracle model where the ciphertext size is $\mathcal{O}(S)$. The public key size is $\mathcal{O}(n)$, while user secret keys and decryption time are constant. Libert et al. [38] proposed a scheme with the same asymptotic performance but secure in the standard model.

Recently, a scheme with public key size $\mathcal{O}(n)$, secret key size $\mathcal{O}(n)$, ciphertext size $\mathcal{O}(r\mathsf{log}(\frac{n}{r}))$, where r is the set $n - S$, and constant decryption time was proposed in [23]. Despite the fact that in this scheme ciphertexts do not grow linearly with n, actually $\mathcal{O}(r\mathsf{log}(\frac{n}{r}))$ is asymptotically larger than $\mathcal{O}(n - r)$ for large values of r, which are likely in our FPLS. Furthermore, the scheme in [23] does not provide anonymity with respect to users who are able to decrypt, i.e., those users can find out the identity of the other users who can decrypt.

We modify the scheme in [3] so that it employs as building block an anonymous identity-based encryption scheme instead of a key-private public key encryption scheme. This allows users to employ their identities as public keys. Such modification was suggested in Barth et al. [3] and security follows from the security of the original scheme.

An identity-based encryption (IBE) scheme consists of the algorithms $(\mathsf{IBESetup}, \mathsf{IBEExtract}, \mathsf{IBEEnc}, \mathsf{IBEDec})$. The algorithm $\mathsf{IBESetup}(1^\lambda)$ outputs parameters $params$ and master secret key msk. $\mathsf{IBEExtract}(params, msk, id)$ outputs the secret key sk_{id} for identity id. $\mathsf{IBEEnc}(params, id, m)$ outputs ct encrypting m under id. $\mathsf{IBEDec}(params, sk_{id}, ct)$ outputs message m encrypted in ct.

An IBE scheme is *anonymous* [1] if it is not possible to associate the identity used to encrypt a message m with the resulting ciphertext. We employ the scheme in [7], which is anonymous [9], to implement the anonymous broadcast encryption scheme.

Another building block of the anonymous IBBE scheme is a strongly existentially unforgeable signature scheme. A signature scheme consists of algorithms $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$. $\mathsf{Kg}(1^\lambda)$ outputs a key pair (ssk, vsk). $\mathsf{Sign}(ssk, m)$ outputs a signature s on message m. $\mathsf{Vf}(vsk, s, m)$ outputs accept if s is a valid signature on m and reject otherwise. We employ the scheme secure in the random oracle model proposed in [6].

The remaining building block is a secure symmetric key encryption scheme (E, D). We employ the advanced encryption standard [19]. The anonymous IBBE scheme works as follows.

$\mathsf{Setup}(1^\lambda, n, \ell)$**.** Choose a group $\mathbb{G}$ of primer order p where CDH is hard and DDH is easy and pick a generator $g \in \mathbb{G}$. Choose a hash function $H : \mathbb{G} \rightarrow \{0,1\}^\lambda$ which is modeled as a random oracle. Compute $params$ and msk via $\mathsf{IBESetup}(1^\lambda)$. For $i = 1$ to n, pick random $a_i \leftarrow \mathbb{Z}_p$. Output Output $pk = (\mathbb{G}, g, g^{a_1}, \ldots, g^{a_n}, H, params)$ and $sk = (msk, a_1, \ldots, a_n)$.

$\mathsf{KeyGen}(i, sk)$**.** On input an index i and the secret key sk, compute a secret key $sk_i \leftarrow \mathsf{IBEExtract}(params, msk, i)$. Output $d_i = (sk_i, a_i)$.

$\mathsf{Enc}(pk, S, m)$**.** On input the recipient group $S \in [1, n]$, the public key pk and the message m, execute the following steps.

1. Compute $(ssk, vsk) \leftarrow \mathsf{Kg}(1^\lambda)$.
2. Pick a random symmetric key K.
3. Pick random $r \leftarrow \mathbb{Z}_p$ and set $T = g^r$.
4. For each tuple $(i, g^{a_i}) \in S$, set the ciphertext $c_i \leftarrow H(g^{a_i r})||\mathsf{IBEEnc}(params, i, vsk||g^{a_i r}||K)$.
5. C_1 is the concatenation of all c_i ordered by the values of $H(g^{a_i r})$.
6. Compute $C_2 = \mathsf{E}_K(m)$.
7. Sign $s \leftarrow \mathsf{Sign}(ssk, T||C_1||C_2)$.
8. Return the ciphertext $C = s||T||C_1||C_2$.

$\mathsf{Dec}(pk, d_j, c)$**.** On input the public key pk, the secret key d_j and a ciphertext c, execute the following steps.

1. Calculate $l = H(T^{a_j})$.
2. Find c_j such that $c_j = l||c$ for some c_j in C_1.
3. Calculate $p \leftarrow \mathsf{IBEDec}(params, sk_j, c_j)$.
4. If p is $\perp$, return $\perp$.
5. Parse p as $vsk||x||K$.
6. If $x \neq T^{a_j}$, return $\perp$.
7. If $\mathsf{Vf}(vsk, s, T||C_1||C_2)$ outputs accept, then output $m = \mathsf{D}_K(C_2)$ else $\perp$.

We remark that, if the user is not in the recipient group and therefore she cannot decrypt, the decryption algorithm only requires the computation of a hash function.

5. PERFORMANCE ANALYSIS

Location-sharing-based applications are usually run on a mobile device, such as a smart phone or a tablet computer. Therefore, the available resources at the client side are limited in terms of computational power and bandwidth when on mobile connection. Furthermore, mobile devices usually have a rather low battery capacity. Thus an application must use the CPU or mobile communication interfaces, such as WiFi or 3G, as moderate as possible in order not to drain the battery too much. In order to evaluate the overhead of our schemes, we implemented them in the C programming language using the *Pairing-Based Cryptography*[2] (PBC) library. Subsequently, we imported the schemes within Android application using Android's Native Development Kit[3] (NDK) and tested the application on a Samsung S III mini (1 GHz dual-core CPU) which runs the CyanogenMod[4] operating system.

In the following we provide the *additional* overhead imposed by our schemes. This overhead is due to the computation of cryptographic operations and due to the transmission of key material and ciphertexts. Mobile applications for LSBSs, such as Foursquare, are used on a large user base and the overhead imposed by these services is accepted in practice.

For the energy consumption of our schemes, we measure the different current consumption of the phone when the CPU is idle and when the CPU load of one core is at the maximum. We found that the difference is 150 mA at 3.8 V and thus that the power consumption is 570 mW if one CPU core is at full load. Please note that we could not use PowerTutor[5] to estimate the energy consumption of our schemes, because PowerTutor was designed for a Nexus 1 mobile phone. Although PowerTutor does also run on our Samsung S III mini, the energy measurements are likely to be inaccurate, because both phones have a different CPU and we found that PowerTutor is unable to read the traffic sent and received on our phone. For the runtime estimation of our schemes, we executed our protocols 50 times and computed the average. Multiplying the runtime with the power consumption equals to the energy consumption of our schemes.

The energy consumption of data transmission via a mobile interface, such as WiFi and 3G, turns out to be significantly more difficult. This is because the actual energy consumption for sending and receiving data depends on many factors, such as amount of data, time between two consecutive data transmissions and network reception. We therefore use Balasubramanian et al. [2] work to outline ways to minimize the energy consumed by our schemes.

5.1 Evaluation of the Sender-Private LSBS

The complexity of the Gentry and Waters [28] scheme that we employ for our SPLS only depends on the size of the maximal broadcast group l. This means that for a given l, the computational and bandwidth overhead for computing the symmetric key stays constant, regardless of n (the number of users in the system) or k (actual receiver of a broadcast message). We therefore limit our evaluation to the system parameter l.

As we can see in Figure 2a the time for creating the symmetric key increases polynomially with l. For a reasonably large l, such as 100, our phone needs 8.66 seconds to compute the symmetric key in the encryption protocol and 7.42 in the decryption protocol. While this appears to be rather high, we must stress that a user is able to reuse a symmetric key until the broadcast group changes or the key got compromised. Therefore, a single symmetric key can be used for thousands of location shares. Furthermore, as we can see in Figure 2b, the actual energy consumed for computing a symmetric key is 4.94 Joule for the encryption protocol and 4.23 Joule for the decryption protocol, which is very low. The capacity of the standard battery of a Samsung S III mini (3.8 V and 1500 mAh) is 20520 Joule and therefore computing even dozens of symmetric keys per day would not drain the battery too much.

We show the bandwidth overhead of the FPLS in Figure 2c. For creating a new symmetric key a user needs to send 788 byte of data to the receiver of the broadcast group. Please note that even for $k > 1$ the user only needs to send 788 byte instead of $k \times 788$ byte, because the service provider forwards the traffic to the intended receivers. The public key of our scheme grows linearly in l. However, please note that the public key only needs to be sent very rarely. This is when a user signs up for the service, the new user receives the public key from the service provider, and if the service provider decides to increase/decrease the size of the maximal broadcast group and thus changes the parameter l. In those cases, all the users in the system receive the new public key.

5.2 Evaluation of the Fully-Private LSBS

In the following we will first show that the computational overhead that is imposed by the FPLS is feasible for current mobile devices. Subsequently we will show that the FPLS imposes a significant bandwidth overhead. This is a problem for two reasons. Firstly, data plans usually include a fixed data volume to be transmitted before either the speed of the connection gets throttled or additional costs incur. Secondly, using mobile communication interfaces, such as 3G or WiFi, is expensive in terms of energy and therefore sending and receiving higher amounts of data additionally drains the battery. However, we make several suggestions on how our protocol should be deployed to significantly reduce the energy consumption, although the transferred data volume remains the same. Furthermore, as introduced in Section 4.3, the concept of big regions greatly helps in making the FPLS feasible. We therefore consider the FPLS as a protocol which allows a very broad-ranged trade-off between location privacy and performance for users that do not wish to reveal their friendship graph. At the one extreme a very high level of location privacy is possible if the user is willing to spend the necessary resources. On the other extreme a user reveals accurate location information towards the service provider and thereby decreases the amount of data that is received.

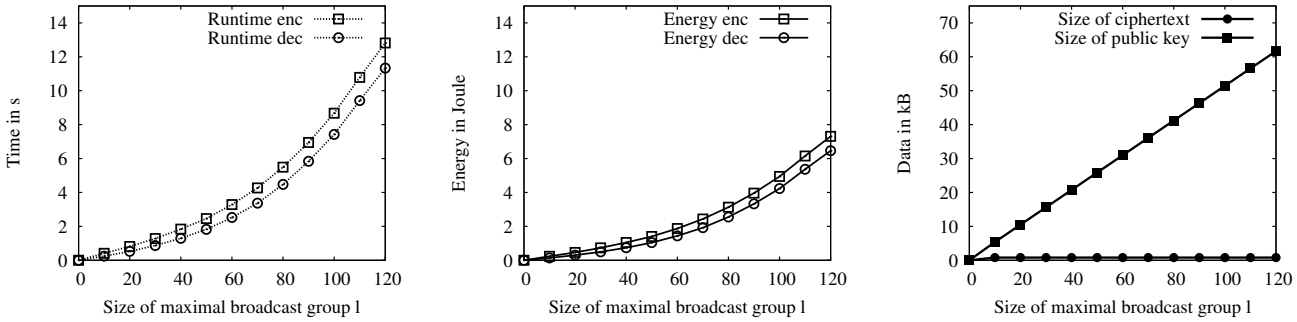
As we can see in Figure 3a, the computation and energy overhead for encryption grows only linearly in k and is there-

[2] http://crypto.stanford.edu/pbc/

[3] https://developer.android.com/tools/sdk/ndk/index.html

[4] http://www.cyanogenmod.org/

[5] http://powertutor.org/



(a) Overview of runtimes for encryption and decryption.

(b) Overview of energy consumption for encryption and decryption.

(c) Overview of bandwidth overhead.

Figure 2: Runtime, energy consumption and bandwidth overhead of the SPLS for increasing l and fixed $n = 1000$ and $k = 5$.

fore lower than in the SPLS. Figure 3b shows that the overhead for decryption is about two magnitudes lower than in the SPLS. Furthermore, Figure 3b shows the computation and energy overhead for fail-decryptions. These are the resources a user needs to spend in order to determine that a location update is actually not for her. Recalling the battery capacity of 20520 Joule, we can see that in terms of computational overhead the FPLS allows for sharing multiple locations per day and receiving thousands of location updates.

As already mentioned the main drawback of the FPLS is the data a user receives due to flooding. Figure 3c shows the lengths of a ciphertext as k increases. In [16], the authors state that the very majority of users has less than 100 friends. Therefore, we can assume most ciphertexts to have a length of at most 60 kB. If an LSBS would have 1 million users, this would result in approximately $10^6 \times 60\text{kB} = 57.22$ GB of data every user receives per day. Clearly, this is not feasible. However, choosing a big region such that only 0.01% of all the world-wide location updates are received results in a bandwidth overhead of 5.86 MB per day.

The energy that is consumed for receiving data can be optimized with two measures. Firstly, receiving data via the WiFi interface consumes significantly less energy than receiving the same amount of data via 3G interface [2] (§ 3.6). For example receiving 500 kB of data via 3G consumes around 19 Joule and only 5 Joule via WiFi. Secondly, as noted in [2] (§ 3.6.1), the energy consumption strongly depends on the inter-transfer time between downloads. For example, receiving 50 kB transmissions with inter-transfer time 1 second consumes around 5 Joule, while it consumes 10 Joule if the inter-transfer time is 9 seconds. We therefore suggest: (i) to receive most of the traffic when a WiFi connection is available; (ii) that the service provider caches location updates for a certain period and sends them all at once in order to have few but large data transmissions to the users.

6. LSBS WITH AGGREGATE STATISTICS COLLECTION

In our scheme, the service provider acts as a channel between users. The service provider relays messages between the sender and the receivers, but learns nothing about the content of the messages sent. The business model of currently deployed LSBS relies on learning user check-ins. Service providers use that information in order to, e.g., give recommendations on most visited places and give discounts to users that check-in a number of times at a particular location.

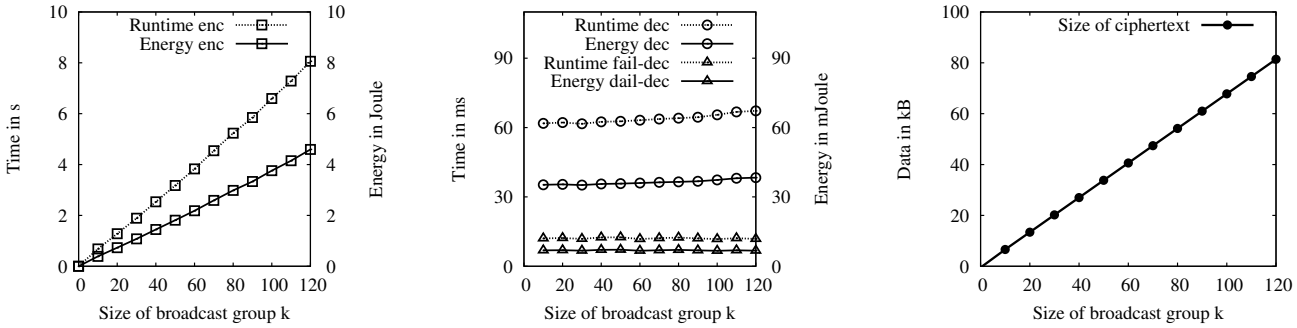
We describe how our scheme can be extended to allow the service provider to collect aggregated data on how many times users visit each of the locations. In this extension, each time a user checks-in, the user increases a committed counter for that location. This committed counter is hidden from the service provider. However, after a number of check-ins, the user can choose to disclose the counter of one of the locations in order to, e.g., get a discount. The commitment ensures that the user cannot cheat and open the committed counter to a different value.

We note that, in currently deployed LSBS, users can check-in at a certain location without being present at that location. The countermeasure against that is that the wrong location disclosed to the service provider is also disclosed to the user's friends, which can cause annoyance.

Our extension provides the same countermeasure. Using zero-knowledge proofs, the user proves to the service provider that the location shared with her friends equals the location whose committed counter is increased. In order to do that, we make the following modification in our scheme. Instead of employing symmetric key encryption to encrypt the location message, we employ public key encryption. The key output when decrypting the broadcast encryption ciphertext is an ElGamal private key, while the corresponding public key is transmitted along with the broadcast encryption ciphertext.

In [16], a scheme that employs hardware devices at the physical location in order to ensure that users can check-in only if they are at the location is proposed. It is also possible to extend our scheme with hardware devices to achieve that property.

We note that the total number of locations where a user can check-in is usually large. The user should maintain a committed counter for each of the locations and, at each check-in, increment it without disclosing the location or the value of the counter, yet proving that the location equals the location shared with her friends. If we employ Pedersen commitments, this operation would have a cost linear on the total number of locations, which would make it impractical. In order to have a cost independent of the total number of



(a) Overview of runtime and energy consumption for encryption.

(b) Overview of runtime and energy consumption for decryption and fail-decryption.

(c) Overview of bandwidth overhead.

Figure 3: Runtime, energy consumption and bandwidth overhead of the FPLS for increasing k and fixed $n = 1000$.

locations, we employ P-commitments [33], which are based on vector commitments.

6.1 Cryptographic Building Blocks

We recall the notation for zero-knowledge proofs and the definitions of P-commitments in [33].

6.1.1 Zero-Knowledge Proofs of Knowledge

We employ of classical results for efficient zero-knowledge proofs of knowledge (ZKPK) for discrete logarithm relations [46, 17, 15, 11, 10, 18]. In the notation of [13], a protocol proving knowledge of exponents $w_1, \ldots, w_n$ satisfying the formula $\phi(w_1, \ldots, w_n)$ is described as

$$\text{ℵ}\, w_1, \ldots, w_n : \phi(w_1, \ldots, w_n) \tag{1}$$

Here, we use the symbol "ℵ" instead of "∃" to indicate that we are proving "knowledge" of the exponents, rather than just its existence. The formula $\phi(w_1, \ldots, w_n)$ consists of conjunctions and disjunctions of "atoms". An atom expresses *group relations*, such as $\prod_{j=1}^{k} g_j^{\mathcal{F}_j} = 1$, where the g_j are elements of prime order groups and the $\mathcal{F}_j$'s are polynomials in the variables $w_1, \ldots, w_n$.

There exist practical zero-knowledge proofs for the types of relations required in our protocols. We refer to Camenisch et al. [12, 14] for details.

Extended zero-knowledge formulas.

A proof system for (1) can be transformed into a proof system for the following more expressive statements about secret exponents $(w_i)_i = \mathit{sexps}$ and secret bases $(g_i)_i = \mathit{sbases}$:

$$\text{ℵ}\, \mathit{sexps}, \mathit{sbases} : \phi(\mathit{sexps}, \mathit{bases} \cup \mathit{sbases}) \tag{2}$$

The transformation uses a blinded base $g_i' = g_i h^{\rho_i}$ for every g_i. It adds h and all g_i' to the public *bases*, ρ_i to the secret *sexps*, and rewrites $g_i^{\mathcal{F}_j}$ into $g_i'^{\mathcal{F}_j} h^{-\mathcal{F}_j \rho_i}$ for all i, j. Finally, we observe that the proof system supports pairing product equations $\prod_{j=1}^{k} e(g_j, \tilde{g}_j)^{\mathcal{F}_j} = 1$ in groups of prime order $|\mathbb{G}|$ with a bilinear map $e : \mathbb{G} \times \tilde{\mathbb{G}} \to \mathbb{G}_t$, by treating the target group as the group of the proof system—we focus on the special case of $i = j$ for simplicity: the embedding for secret bases is unchanged, except for the case in which both bases in a pairing are secret. In the latter case, $e(g_j, \tilde{g}_j)^{\mathcal{F}_j}$ needs to be transformed into $e(g_j', \tilde{g}_j')^{\mathcal{F}_j} e(g_j', \tilde{h})^{-\mathcal{F}_j \rho_j} e(h, \tilde{g}_j')^{-\mathcal{F}_j \tilde{\rho}_j} e(h, \tilde{h})^{-\mathcal{F}_j \rho_j \tilde{\rho}_j}$.

Macro notation for zero-knowledge statements.

We use a macro language to specify named proof components that can be reused as sub-components of larger proofs. For example, we may define a proof macro for the long division of a by q as follows: $\mathbf{Div}(a, q) \mapsto (b) \equiv \text{ℵ}\, a, q, b, r : a = b * q + r \wedge r < b \wedge 0 \leq a, b, q, r \leq \sqrt{|\mathbb{G}|} \wedge b > 1$. Semantically, the function **Div** states that the division of a by q gives b with remainder r. Secret a is interpreted as an initial value and secret b as a new value. In terms of cryptography, it is simply syntactic sugar, but important sugar as demonstrated by the long list of side conditions to guarantee a unique positive solution modulo the order of $\mathbb{G}$. Proving these inequalities is itself non-trivial and could be further expanded using macros.

Named proof components can be used in further higher-level proofs without their details being made explicit. For example, the proof $\text{ℵ} \ldots, a, q, b : \cdots \wedge b = \mathbf{Div}(a, q)$ can from now on be used in proof expressions instead of the complex restrictions above. All variables within the component declaration (e.g., variables a, q, b in $\mathbf{Div}(a, q) \mapsto (b)$) can be re-used in the high level proof. Variables whose knowledge is proved, but that are not in the declaration, are considered inaccessible to the higher-level proof.

6.1.2 P-commitments

A vector commitment scheme allows Alice to succinctly commit to a vector $\mathbf{x} = \langle x_1, \ldots, x_n \rangle \in \mathcal{M}^n$ such that she can compute an opening w to x_i and can replace x_i by a new value x_i' by updating her commitment, such that both w and the update value is of size independent of i and n. A vector commitment scheme consists of the following algorithms.

$\mathsf{Setup}(1^k, \ell)$**.** On input the security parameter 1^k and an upper bound ℓ on the size of the vector, generate the parameters of the commitment scheme par, which include a description of the message space $\mathcal{M}$ and a description of the randomness space $\mathcal{R}$.

$\mathsf{Commit}(par, \mathbf{x}, r)$**.** On input a vector $\mathbf{x} \in \mathcal{M}^n$ $(n \leq \ell)$ and $r \in \mathcal{R}$, output a commitment com to $\mathbf{x}$.

$\mathsf{Prove}(par, i, \mathbf{x}, r)$**.** Compute a witness w for x_i.

$\mathsf{Verify}(par, com, x, i, w)$**.** Output accept if w is a valid witness for x being at position i and reject otherwise.

$\mathsf{Update}(par, com, i, x, r, x', r')$. On input a commitment com with value x at position i and randomness r, output a commitment com' with value x' at position i and randomness r'. The other positions remain unchanged.

A commitment scheme must be hiding and binding. The hiding property requires that any probabilistic polynomial time (p.p.t.) adversary $\mathcal{A}$ has negligible advantage in guessing which of two vectors $\mathbf{x}$ of values of its choice has been used to compute a challenge commitment. The binding property requires that any p.p.t. adversary $\mathcal{A}$ has negligible advantage in computing a vector $\mathbf{x}$ of length n, randomness r, a value x, a position $i \in [1..n]$ and a witness w such that $\mathbf{x}[i] \neq x$ but the commitment $com \leftarrow \mathsf{Commit}(par, \mathbf{x}, r)$ can be opened to x at position i using w.

A P-commitment scheme is a secure vector commitment scheme that supports three ZKPKs.

Create. A proof of correct commitment generation that proves knowledge of $(\mathbf{x}, r)$ such that $\mathsf{Commit}(par, \mathbf{x}, r) = com$. We call the proof macro $\mathbf{Create}(\mathbf{x}) \rightarrow (com, r, (w_i)_i)$ as it proves that a P-commitment was correctly initialized to the vector $\mathbf{x}$. The prover can then use this commitment in subsequent proof steps. To simplify our macro notation, we use $M = (com, r, (w_i)_i)$ as a shorthand for the collection of com, r, and different w_i values and refer to it as the memory of a P-commitment proof.

Get. A proof of a witness w that a value x was committed to in com at position i.

$$\begin{aligned}&\mathbf{Get}(M, i) \rightarrow (x) \equiv \\ &\quad \nvDash x, i, w : \\ &\quad \mathsf{Verify}(par, com, x, i, w) = \mathsf{accept} \ \wedge i \in [1, n]\end{aligned}$$

Set. A proof that a commitment com' is an update of commitment com at position i. This proof is slightly more involved because it requires the prover to prove knowledge of the old vector value for the updated position to bind the old and the new commitment together:

$$\begin{aligned}&\mathbf{Set}(M, i, x') \rightarrow (M') \equiv \\ &\quad \nvDash \mathbf{x}[i], x', i, r, r', w : \\ &\quad com' = \mathsf{Update}(par, com, i, \mathbf{x}[i], r, x', r') \ \wedge \\ &\quad \mathsf{Verify}(par, com, \mathbf{x}[i], i, w) = \mathsf{accept} \wedge i \in [1, n]\end{aligned}$$

6.2 Construction

As mentioned above, in this extension the location message m is encrypted using an ElGamal encryption $c = (c_1, c_2) = (y^r \cdot m, g^r)$, where $y = g^x$ is the public key and x is the secret key. The secret key is encrypted in the broadcast encryption ciphertext, while the public key is transmitted along with the broadcast encryption ciphertext. We employ a zero-knowledge proof of knowledge of m encrypted to in c:

$$\nvDash m, t : e(c_1, g) = e(m, g) \cdot e(t, g) \wedge e(t, g) = e(y, c_2)$$

In our scheme, the indices $(i_1, \ldots, i_n)$ of the committed vector will be the locations, and n is the total number of locations. We note that the schemes proposed in [33] to implement P-commitments employ a structure preserving signature (SPS) scheme to sign together an index i with the generator g_i for position i in the parameters of the commitment scheme. SPS sign group elements, and therefore we can prove in zero-knowledge equality between the location message m encrypted in c and the index i of the P-commitment. Sender and receivers employ a table or a hash function to map a location to an element of group $\mathbb{G}$.

In the registration phase, the service provider executes $\mathsf{Setup}(1^k, \ell)$, where ℓ is the number of locations, and sends par to the users. Then, each user creates a vector $\mathbf{x} = (0, \ldots, 0)$ of size ℓ, picks $r \in \mathcal{R}$ and runs $\mathsf{Commit}(par, \mathbf{x}, r)$ to get com. The user sends com to the service provider, along with a proof

$$\nvDash \mathbf{x} : \mathbf{Create}(\mathbf{x}) \rightarrow (M) \wedge \mathbf{x} = (0, \ldots, 0)$$

where $M = (com, r, (w_i)_i)$. This proof initializes the counters for each of the locations to 0 and can be done very efficiently in the P-commitment schemes in [33].

In the main phase, when a user sends a broadcast ciphertext to the service provider for location i encrypted in ciphertext (c_1, c_2), the user sets $\mathbf{x}[i]' = \mathbf{x}[i] + 1$, picks random $r' \in \mathcal{R}$ and runs $\mathsf{Update}(par, com, i, \mathbf{x}[i], r, \mathbf{x}[i]', r')$ to get com'. The user sends com' to the service provider, along with a proof

$$\begin{aligned}&\nvDash i, t, \mathbf{x}[i], \mathbf{x}[i]' : \\ &\quad e(c_1, g) = e(i, g) \cdot e(t, g) \wedge e(t, g) = e(y, c_2) \wedge \\ &\quad \mathbf{Set}(M, i, \mathbf{x}[i]') \rightarrow (M') \wedge \mathbf{x}[i]' = \mathbf{x}[i] + 1\end{aligned}$$

The user proves that she increments the committed counter for the same location in the message encrypted in c. We recall that the cost of this proof is independent of the number of locations. When the service provider receives the broadcast encryption ciphertext, the ElGamal ciphertex and public key, the commitment com' and the proof, the provider verifies the proof. If it is correct, then the provider replaces the stored com by com' and sends the broadcast encryption ciphertext and the ElGamal ciphertext and public key to the receivers. The receivers decrypt first the broadcast encryption ciphertex to get the ElGamal secret key, which is used to decrypt the ElGamal ciphertext and get the sender's location.

When a user wishes to open the counter corresponding to one of the locations, she can use algorithm $\mathsf{Prove}(par, i, \mathbf{x}, r)$ to compute a witness w for location i and send $(\mathbf{x}[i], i, w)$ to the provider. The service provider runs $\mathsf{Verify}(par, com, \mathbf{x}[i], i, w)$ and accepts the value of the counter $\mathbf{x}[i]$ if the verification is successful. Alternatively, the user can also prove statements about the committed counter in zero-knowledge, e.g., prove that the counter has surpassed a threshold that entitles her to receive a discount. The proof **Get** is employed for this purpose.

The security of this extension relies on the security of P-commitments. The hiding property, along with the zero-knowledge property of proofs of knowledge, ensures that the service provider does not learn the values of the committed counters or the locations whose counters are increased. Additionally, the binding property of P-commitments and the extractability of proofs of knowledge ensure that the committed counters are updated correctly and that they cannot be opened to a wrong value.

7. DISCUSSION

The computation overhead of the SPLS is mainly due to the creation of a symmetric key. An actual location sharing operation is then encrypted using a fast encryption scheme,

such as AES. While we note that the symmetric key of the SPLS can be reused for many location sharing operations, we argue that even computing several symmetric keys per day is feasible. Firstly, as our evaluation shows computing symmetric keys consumes very little energy and can thus be done several times without draining the battery. Secondly, since modern smart phones have multi-core processors embedded, one core can be occupied for creating a symmetric key while the phone is still usable for other operations, such as email checking or surfing the web. All in all we thus argue that on the user side our scheme imposes negligible overhead to the user's device.

The FPLS on the other hand imposes a significant bandwidth overhead. However, we note that it is to the best of our knowledge the only scheme that allows a user to hide her friends without relying on proxies, such as in [45]. It does so by offering a privacy/performance trade-off, which has been proposed before in schemes for privacy-preserving LSBS [45]. Note that the FPLS is not vulnerable to velocity-based attacks [29] for two reasons. Firstly location updates happen sporadically and not continuously and hence big regions are much harder to correlate. Secondly, and more importantly, the big regions are much bigger than in k-anonymity schemes, such as [31].

We note that our schemes are also suitable to implement other services, such as social recommendation applications. This is because in practice users can share arbitrary information in the ciphertext. Instead of encrypting location information, users could share their reviews, such as how they like the food in a particular restaurant. Furthermore, the low overhead of the SPLS and the strategy of reusing symmetric keys would allow to use the scheme for location tracking applications. Such applications require rather frequent location updates instead of sporadic ones, which is usually the case for check-in applications. Furthermore, we note that our schemes are more efficient than a unicast solution in which every user sends an encrypted location update to each of her friends. This is because in our schemes the user only needs to transmit the encrypted location update to the service provider that is then forwarding it to the recipients or all users of the service, respectively. This consumes significantly less bandwidth and also less energy than in the unicast solution.

Although in the setup routine of the SPLS, as well as the FPLS, the service provider initially needs to commit to a maximal number of users n, we note that even if there are more than n users in the service, the service provider does not need to re-initialize the service. In the SPLS, n is only used for checking that $l \leq n$. This condition, however, is maintained if it was true before and n increases. In the FPLS, the scheme's public key can be extended, because the g^{a_i} with $i > n$ can be computed when necessary and distributed among all users of the service.

Besides location-sharing, badge and mayorship protocols are another main functionality of a GSN. For the latter privacy preserving protocols have been proposed in [16]. We note that it would be possible to combine both approaches to build a privacy-preserving GSN.

8. CONCLUSIONS

We have defined the privacy properties that an LSBS should provide and we have proposed two LSBS based on identity-based broadcast encryption. Both constructions allow a user to share her location with her friends without disclosing it to the service provider. The first construction discloses to the service provider the receivers of a location update, while the second does not. As advantages from previous work, in our schemes the LSBS provider does not need to perform complex operations in order to compute a reply for a location data request, but only needs to forward IBBE ciphertexts to the receivers. This allows to run a privacy-preserving LSBS at significantly lower costs. We implement both constructions and present a performance analysis that shows their practicality. Furthermore, we extend our schemes such that the service provider, performing some verification work, is able to collect privacy-preserving statistics about the places users share among each other. This could be a way to monetize the privacy-preserving LSBS.

Acknowledgments.

We thank the anonymous reviewers for their valuable comments. We also thank Carmela Troncoso for insightful discussions and helpful feedback. This research was supported in part by the projects: IWT SBO SPION, FWO G.0360.11N, FWO G.0686.11N, GOA TENSE (GOA/11/007), iMinds SoLoMIDEM, and KU Leuven OT project Traffic Analysis Resistant Privacy Enhancing Technologies.

9. REFERENCES

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *Advances in Cryptology–CRYPTO 2005*, pages 205–222. Springer, 2005.

[2] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 280–293. ACM, 2009.

[3] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. *Financial Cryptography and Data Security*, pages 52–64, 2006.

[4] A.R. Beresford and F. Stajano. Mix zones: User Privacy in Location-aware Services. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 127 – 131, march 2004.

[5] Igor Bilogrevic, Murtuza Jadliwala, Kübra Kalkan, Jean-Pierre Hubaux, and Imad Aad. Privacy in mobile computing for location-sharing-based services. In *Privacy Enhancing Technologies*, volume 6794 of *Lecture Notes in Computer Science*, pages 77–96. Springer Berlin Heidelberg, 2011.

[6] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology-EUROCRYPT 2004*, pages 56–73. Springer, 2004.

[7] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology-CRYPTO 2001*, pages 213–229. Springer, 2001.

[8] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology–CRYPTO 2005*, pages 258–275. Springer, 2005.

[9] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology-CRYPTO 2006*, pages 290–307. Springer, 2006.

[10] Stefan Brands. Rapid demonstration of linear relations connected by boolean operators. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *LNCS*, pages 318–333. Springer Verlag, 1997.

[11] Jan Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zürich, 1998.

[12] Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential authenticated identification and key exchange. In *CRYPTO*, pages 255–276, 2010.

[13] Jan Camenisch, Stephan Krenn, and Victor Shoup. A framework for practical universally composable zero-knowledge protocols. In *ASIACRYPT*, pages 449–467, 2011.

[14] Jan Camenisch, Stephan Krenn, and Victor Shoup. A framework for practical universally composable zero-knowledge protocols. Cryptology ePrint Archive, Report 2011/228, 2011. `http://eprint.iacr.org/`.

[15] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number n is the product of two safe primes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *LNCS*, pages 107–122. Springer Verlag, 1999.

[16] Bogdan Carbunar, Radu Sion, Rahul Potharaju, and Moussa Ehsan. The shy mayor: Private badges in geosocial networks. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *Applied Cryptography and Network Security*, volume 7341 of *Lecture Notes in Computer Science*, pages 436–454. Springer Berlin Heidelberg, 2012.

[17] D. Chaum and T. Pedersen. Wallet databases with observers. In *CRYPTO '92*, volume 740 of *LNCS*, pages 89–105, 1993.

[18] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, pages 174–187, 1994.

[19] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer, 2002.

[20] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. *Advances in Cryptology–ASIACRYPT 2007*, pages 200–215, 2007.

[21] Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. *Public Key CryptographyâĂŤPKC 2003*, pages 100–115, 2002.

[22] Changyu Dong and Naranker Dulay. Longitude: A privacy-preserving location sharing protocol for mobile applications. In *Trust Management V*, volume 358 of *IFIP Advances in Information and Communication Technology*, pages 133–148. Springer Berlin Heidelberg, 2011.

[23] Nelly Fazio and Irippuge Perera. Outsider-anonymous broadcast encryption with sublinear ciphertexts. *Public Key Cryptography–PKC 2012*, pages 225–242, 2012.

[24] Amos Fiat and Moni Naor. Broadcast encryption. In *Advances in CryptologyâĂŤCryptoâĂŹ93*, pages 480–491. Springer, 1994.

[25] J. Freudiger, R. Shokri, and J. Hubaux. Evaluating the Privacy Risk of Location-Based Services. In *Financial Cryptography and Data Security*, volume 7035 of *LNCS*, pages 31–46. Springer Berlin, 2012.

[26] Julien Freudiger, Raoul Neu, and Jean-Pierre Hubaux. Private sharing of user location over online social networks. *3rd Hot Topics in Privacy Enhancing Technologies (HotPETs 2010)*, 2010.

[27] Sébastien Gambs, Olivier Heen, and Christophe Potin. A comparative privacy analysis of geosocial networks. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '11, pages 33–40, New York, NY, USA, 2011. ACM.

[28] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). *Advances in Cryptology-EUROCRYPT 2009*, pages 171–188, 2009.

[29] Gabriel Ghinita, Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. Preventing velocity-based linkage attacks in location-aware applications. GIS '09, pages 246–255, New York, NY, USA, 2009. ACM.

[30] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private queries in location based services: anonymizers are not necessary. SIGMOD '08, pages 121–132, New York, NY, USA, 2008. ACM.

[31] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. MobiSys '03, pages 31–42, New York, NY, USA, 2003. ACM.

[32] Leping Huang, Hiroshi Yamane, Kanta Matsuura, and Kaoru Sezaki. Towards modeling wireless location privacy. In George Danezis and David Martin, editors, *Privacy Enhancing Technologies*, volume 3856 of *Lecture Notes in Computer Science*, pages 59–77. Springer Berlin Heidelberg, 2006.

[33] Markulf Kohlweiss and Alfredo Rial. Optimally private access control. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 37–48. ACM, 2013.

[34] John Krumm. Inference attacks on location tracks. In Anthony LaMarca, Marc Langheinrich, and KhaiN. Truong, editors, *Pervasive Computing*, volume 4480 of *Lecture Notes in Computer Science*, pages 127–143. Springer Berlin Heidelberg, 2007.

[35] John Krumm. Realistic driving trips for location privacy. In Hideyuki Tokuda, Michael Beigl, Adrian Friday, A. Brush, and Yoshito Tobe, editors, *Pervasive Computing*, volume 5538 of *Lecture Notes in Computer Science*, pages 25–41. Springer Berlin / Heidelberg, 2009.

[36] Allison Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 273–285. IEEE, 2010.

[37] Mingyan Li, Krishna Sampigethaya, Leping Huang, and Radha Poovendran. Swing & swap: User-centric approaches towards maximizing location privacy. WPES '06, pages 19–28, New York, NY, USA, 2006. ACM.

[38] Benoît Libert, Kenneth Paterson, and Elizabeth Quaglia. Anonymous broadcast encryption: adaptive security and efficient constructions in the standard model. *Public Key Cryptography–PKC 2012*, pages 206–224, 2012.

[39] Zi Lin, Denis Foo Kune, and Nicholas Hopper. Efficient private proximity testing with gsm location sketches. In AngelosD. Keromytis, editor, *Financial Cryptography and Data Security*, volume 7397 of *Lecture Notes in Computer Science*, pages 73–88. Springer Berlin Heidelberg, 2012.

[40] Dalit Naor, Moni Naor, and Jeff Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology-CRYPTO 2001*, pages 41–62. Springer, 2001.

[41] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. Location privacy via private proximity testing. In *NDSS*, 2011.

[42] Femi Olumofin, Piotr Tysowski, Ian Goldberg, and Urs Hengartner. Achieving efficient query privacy for location based services. In *Privacy Enhancing Technologies*, pages 93–110. Springer, 2010.

[43] Tatiana Pontes, Marisa Vasconcelos, Jussara Almeida, Ponnurangam Kumaraguru, and Virgilio Almeida. We know where you live: Privacy characterization of foursquare behavior. In *4th International Workshop on Location-Based Social Networks (LBSN 2012)*, 2012.

[44] Raluca Ada Popa, Andrew J Blumberg, Hari Balakrishnan, and Frank H Li. Privacy and accountability for location-based aggregate statistics. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 653–666. ACM, 2011.

[45] K.P.N. Puttaswamy, Shiyuan Wang, T. Steinbauer, D. Agrawal, A. El Abbadi, C. Kruegel, and B.Y. Zhao. Preserving location privacy in geosocial applications. *Mobile Computing, IEEE Transactions on*, 13(1):159–173, Jan 2014.

[46] C. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.

[47] R. Shokri, G. Theodorakopoulos, J. Le Boudec, and J. Hubaux. Quantifying Location Privacy. In *Security and Privacy (SP)*, pages 247 –262, May 2011.

[48] C.R. Vicente, D. Freni, C. Bettini, and Christian S. Jensen. Location-related privacy in geo-social networks. *Internet Computing, IEEE*, 15(3):20–27, 2011.

[49] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. *Advances in Cryptology-CRYPTO 2009*, pages 619–636, 2009.

[50] Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, lester and pierre: Three protocols for location privacy. In Nikita Borisov and Philippe Golle, editors, *Privacy Enhancing Technologies*, volume 4776 of *Lecture Notes in Computer Science*, pages 62–76. Springer Berlin Heidelberg, 2007.